2008/63

■

A tight bound on the throughput of queueing networks with blocking

Jean-Sébastien Tancrez, Philippe Chevalier
and Pierre Semal

CORE
Voie du Roman Pays 34
B-1348 Louvain-la-Neuve, Belgium.
Tel (32 10) 47 43 04
Fax (32 10) 47 43 01
E-mail: corestat-library@uclouvain.be
http://www.uclouvain.be/en-44508.html

CORE DISCUSSION PAPER
2008/63

A tight bound on the throughput of queueing networks with blocking

Jean-Sébastien TANCREZ[1], Philippe CHEVALIER[2]
and Pierre SEMAL [3]

November 2008

Abstract

In this paper, we present a bounding methodology that allows to compute a tight lower bound on the cycle time of fork--join queueing networks with blocking and with general service time distributions. The methodology relies on two ideas. First, probability masses fitting (PMF) discretizes the service time distributions so that the evolution of the modified network can be modelled by a Markov chain. The PMF discretization is simple: the probability masses on regular intervals are computed and aggregated on a single value in the corresponding interval. Second, we take advantage of the concept of critical path, i.e. the sequence of jobs that covers a sample run. We show that the critical path can be computed with the discretized distributions and that the same sequence of jobs offers a lower bound on the original cycle time. The tightness of the bound is shown on computational experiments. Finally, we discuss the extension to split--and--merge networks and approximate estimations of the cycle time.

Keywords: queueing networks, blocking, throughput, bound, probability masses fitting, critical path.

[1] CORE, Université catholique de Louvain and LSM, Facultés Universitaires Catholiques de Mons, Belgium. E-mail: js.tancrez@uclouvain.be
[2] CORE and Louvain School of Management, Université catholique de Louvain, Belgium. E-mail: philippe.chevalier@uclouvain.be. This author is also member of ECORE, the newly created association between CORE and ECARES.
[3] Louvain School of Management, Université catholique de Louvain, Belgium. E-mail: Pierre.semal@uclouvain.be.

The work of J.-S. Tancrez has been funded by the European Social Fund and the Walloon Region of Belgium. It is made in collaboration with ArcelorMittal Fontaine.

This paper presents research results of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsibility is assumed by the authors.

1 Introduction

In this paper, we are interested in the modelling of queueing networks with finite buffers, which means blocking can occur when the next buffer is full. We focus on fork-join queueing networks (FJQN). Other configurations are discussed at the end of the paper. A FJQN is a queueing network in which the nodes are linked arbitrarily without forming loops. The servers can have several input or output servers, but each buffer has exactly one upstream server and one downstream server [12]. When a service ends in a server, one job is taken from each upstream buffer and one job is put in each downstream buffer. An example of FJQN is given in Figure 1. In a fork server (e.g. S_2 in Figure 1), one item is taken from the previous buffer (except if it is empty), disassembled into several pieces, and put in the next buffers (except if they are full). In a join server (e.g. S_6 in Figure 1), items are taken from the previous buffers (except if they are empty), assembled into one unit, and the latter is put in the next buffer (except if it is full). Such queueing systems arise in many applications areas such as manufacturing (assembly/disassembly systems) or parallel processing.

Furthermore, we take a few other assumptions. The service time distributions are general, but assumed to be finite, what is always the case in practice. We suppose the network to be saturated, i.e. with infinite supply and demand (sources, like S_1 and S_3 in Figure 1, are never starved, and sinks, like S_7 and S_8, are never blocked). The blocking policy is supposed to be "blocking after service", i.e. if the next buffer is full when a server ends its job, the job waits in the server. These assumptions are not restrictive as virtual servers could be used to model arrivals or demand, and as the method could be fitted to other blocking policies.

In general, queueing networks with general service time distributions

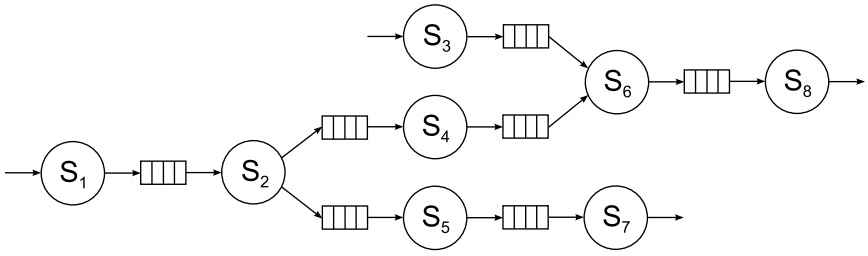

Figure 1: Example of a fork-join queueing network (FJQN).

cannot be analyzed exactly. In order to analytically model them, the distributions have to be transformed to tractable distributions, which means phase–type distributions in most cases. This can be considered as the first step of the global modelling process. Moments fitting is most commonly used (see Whitt [24], Johnson & Taaffe [14] or Bobbio et al. [7] for example). The maximum likelihood estimation (see Asmussen et al. [2] or Bobbio et al. [6]) and the minimum distance estimation (see Varah [23]) are other options [19]. Lang & Arthur [17] conducted a vast experimental study of some fitting methods, based on moments fitting as well as on likelihood maximization. They concluded that a "moderate order (order 4 or order 8)" is satisfactory for the "phase–type behaved" distributions while an higher order is necessary for other distributions (low variance, steep increase or decrease, multiple modes, finite support). In a previous article, we proposed an alternative method to build tractable discrete distributions: probability masses fitting [20]. This method is used in the present paper. It is briefly presented in Section 2.1.

It is important to note that this first step, the transformation of the original distributions, unavoidably introduces an error in the subsequent performance analysis. In other words, any modelling of a queueing network with general distributions is approximate. No exact evaluation can be reached in the general case. However, in the literature, most authors omit this first step of the modelling process. Doing so, they neglect the approximation error coming from the building of tractable distributions. In this context, many analytical models of queueing networks have been proposed [1, 5, 8, 19]. They can be viewed as the second step of the global modelling process (when phase–type distributions have been built). They are of two types: exact or approximate. For non–trivial configurations, the main exact methods are called state models, and consist in the exact modelling of the evolution of the system by a Markov Chain. Their limitation comes from the explosion of the state space size. Consequently, approximate methods have been proposed. The most popular models are based on the idea of decomposing the system into smaller subsystems, and then including back the interdependencies between the subsystems (see, for example, the survey by Dallery & Frein [10] for the decomposition method and Kerbache & Smith [15] for the expansion method).

In this paper, we analyze queueing networks with general distributions (we do not suppose phase–type distributions). In other words, we consider the complete modelling process, including the building of tractable distributions and the proper analytical modelling. As explained earlier, the exact modelling of queueing networks with general distributions is not possible.

In this context, bounds represent the second exact option. It can be used to test approximations or simulation models, or to develop approximations with known accuracy. This paper aims to propose a new lower bound on the cycle time, i.e. the average time in steady–state between two units leaving the system.

The literature proposing bounds on the performance measures of queueing networks can be divided in two types: assuming phase–type distributions or not. Most papers assume phase–type distributions (i.e. focus on the second step of the modelling). In this case, exact solutions can be computed, theoretically. Researchers thus look for approximations and bounds which are quicker to compute. In the following, we cite the main bounding methodologies proposed in the literature. Liu and Buzacott [18] proposed throughput bounds related to the decomposition methods mentioned previously. To get a lower bound on the cycle time (upper bound on the throughput), they increase the capacity of all buffers to infinity, except for the buffer of the first subsystem. Another bounded approximation is known as bounded aggregation [9, 5]. It decomposes the underlying Markov process, i.e. its state space, and derives bounds on the stationary probabilities, with various levels of accuracy and complexity. Kumar and coauthors proposed to use constraints on the behavior of the system to obtain a linear program which leads to upper and lower bounds on the throughput [16]. Another approach, called Performance Bound Hierarchy, has been proposed by Eager and Sevcik [13]. Their recursive algorithm computes bounds whose tightness improves with the number of recursive steps.

Concerning queueing networks with general distributions, few papers have proposed bounds. The first approach relies on the concavity and monotonicity properties of queueing systems [22, 5, 11]. Increasing the buffers capacities leads to lower bounds on the cycle time while decreasing it leads to upper bounds. However, these properties are useful only if the modified network is tractable, which can be the case for zero or infinite buffer capacities. With such an approach, the achieved accuracy is poor, and it is impossible to evaluate the impact of the size of the buffers. A second approach is the bounding methodology developed by Baccelli and coauthors [4], for systems with synchronization constraints and general distributions. It relies on the recursion equation of the system and on the theory of stochastic ordering. Baccelli and Liu applied the methodology to stochastic decision free petri nets, which can model various queueing networks [3]. They show that stochastic ordering of service times leads to stochastic ordering of the cycle time. Unfortunately, the tightness of the bounds has not been investigated. Finally, in previous works [20, 21], we proposed bounds on the through-

put of tandem queues with general service time distributions, using another bounding methodology which leads to less tight bounds.

The rest of the paper is organized as follows. In Section 2, we describe the discretization method we use to get tractable distributions (probability masses fitting), and, then, we present the global modelling method, where the transformed system is modelled by a Markov chain. Section 3 then presents the critical path, its relevant properties and its computation. In Section 4, we show the main result of the paper: a new lower bound on the cycle time. Next, in Section 5, we assess the tightness of the bound by computational experiments on various network structures. In Section 6, we discuss an extension of the proposed bounding methodology to split and merge configurations and we show that an accurate approximation of the cycle time can also be computed. Finally, we conclude in Section 7.

2 Modelling

In this section, we present the global modelling method. Its main originality comes in the first step, i.e. in the building of tractable distributions. We propose an alternative method called probability masses fitting (PMF) to build discrete phase–type distributions. After PMF, the evolution of the queueing network can be modelled by a Markov chain, from which the performances can be evaluated. This modelling method has been introduced and extensively discussed in [20] for tandem queues. In the following, we first briefly present probability masses fitting and, then, the global modelling method.

2.1 Probability Masses Fitting

Probability masses fitting (PMF) is quite intuitive: the probability masses on regular intervals are computed and aggregated on a single value in the corresponding interval. It is illustrated in Figure 2. Formally, PMF transforms a given finite distribution into a discrete one by aggregating the probability mass distributed in the interval $((j-1)\tau+\alpha, j\tau+\alpha]$ on the point $j\tau$, for $j = 2, \ldots, a$, and the mass in $[0, \tau+\alpha]$ on τ. Note that the first interval is treated differently in order to avoid a discrete value on zero, generating jobs of length zero and complicating the resolution. The parameter a gives the number of discrete values, and α gives the shift of the probability mass intervals compared to the corresponding discrete values. The parameter τ gives the interval between two discrete values (which has to be constant), as well as the size of the interval on which the probability mass is computed.

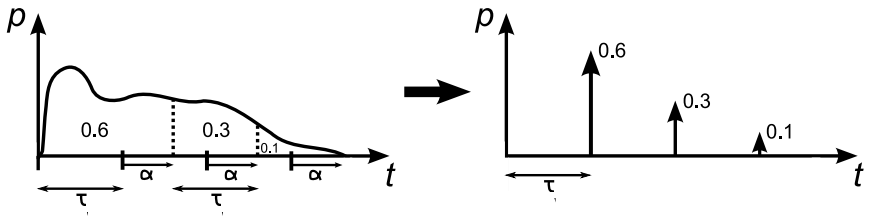

Figure 2: Discretization by probability masses fitting, with $a = 3$.

In the following, the random variable representing the service time of $w_{i,k}$, the job k on server i, is denoted $l(w_{i,k})$. The realization of $l(w_{i,k})$ in a sample run r, i.e. the time the job $w_{i,k}$ takes in this particular run r, is denoted $l^r(w_{i,k})$. The service times discretized by PMF are denoted $l_\alpha(w_{i,k})$ and the realizations $l^r_\alpha(w_{i,k})$. With these notations, the PMF can be formulated as follows:

$$\begin{array}{rcll} l^r_\alpha(w_{i,k}) & \triangleq & \tau, & \text{if } l^r(w_{i,k}) \leq \tau + \alpha, \quad \forall\, r, i, k, \\ & \triangleq & \left\lceil \frac{l^r(w_{i,k})-\alpha}{\tau} \right\rceil \tau, & \text{if } l^r(w_{i,k}) > \tau + \alpha, \quad \forall\, r, i, k. \end{array}$$

Probability masses fitting has some interesting properties. By definition, it is simple, it preserves the shape of the original distribution and it is refinable, i.e. the fitting can be improved by increasing the number a of discretization steps. It is easily seen that $l^r_\alpha(w_{i,k}) - \tau \leq l^r(w_{i,k}) \leq l^r_\alpha(w_{i,k}) + \alpha$. In [20], we show that these bounds on the service time can be extended to bounds on the cycle time, $c_\alpha - \tau \leq c \leq c_\alpha + \alpha$, where c and c_α are the cycle time in the original and in the discretized time, respectively. Finally, the main weaknesses lies in the fact that it is not well suited for service time distributions including rare events.

2.2 Modelling Method

The global modelling method can be described as follows. First, the original, general, service time distributions are transformed to discrete phase–type distributions by probability masses fitting. Note that the step size τ has to be the same for each server's distribution. Then, an analytical model can be used. Various state–of–the art models could be applied (see Section 1). In this paper, aiming to show a bound on the cycle time, we use an exact model, more precisely a state model. The evolution of the system is described by a Markov chain whose states are the possible combinations of

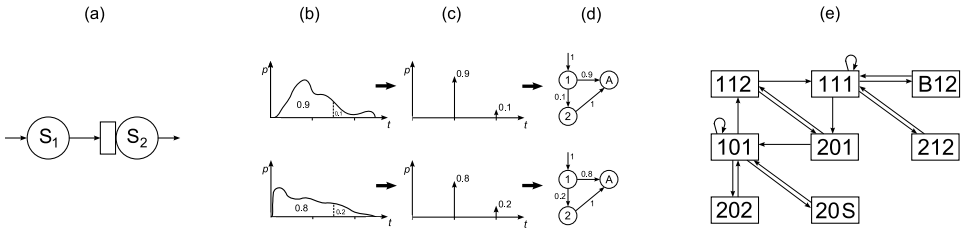

Figure 3: Steps of the modelling method, applied to a two servers tandem queue.

the stages of the various servers and the number of units in the buffers. The performance of the queueing system can then be evaluated from the Markov chain.

The method is best illustrated on a simple example. Let us consider the simplest possible network: a two servers tandem queue (see Figure 3.a). The buffer size equals one. The original service time distributions are shown in Figure 3.b. The first step of the method aims to build tractable distributions. Probability masses fitting is applied with $a = 2$ and $\alpha/\tau = 1/2$, i.e. aggregating the probability mass in the middle of the interval. It leads to the discrete distributions given in Figure 3.c. They can then be represented as discrete phase–type distributions (Figure 3.d). Then, the second step of the method comes into play: the system is analytically modelled. The Markov chain given in Figure 3.e lists all the possible recurrent states of the system and the transitions between these states. The first symbol of a state refers to the first server, the second to the queue and the third to the second server. Each server can be starved (S), blocked (B) or in some stage of service (for example, 1 means that the server already spent one time step serving the current unit). Each queue is described by the number of units waiting in the buffer (0 or 1 in this case). For example, state B12 means that the first server is blocked, that the buffer is full and that the second server already worked during two time steps on the current job. From a state, the transition to a new state depends on a server ending its current job or not. From state 112 for example, in the next time step, the second server will finish its current job, and begin to serve the unit which is waiting in the queue. So, if the first server ends its job, he will put the unit in the queue and begin a new job, leading to state 111. On the other hand, if the first server continues to serve the same unit, the new state will be 201. Finally, the Markov chain can be built applying this logic to every state, and the

performances of the system can be computed from the chain.

The size of the Markov chain is, in first approximation, proportional to the number of combinations of individual servers and queues states, i.e. $\prod_{i=1}^{m}(a_i+2)\cdot\prod_{i=2}^{m}(b_i+1)$ with m the number of servers, a_i the number of discrete values in the discretized service time distribution of server i, and b_i the size of the buffer preceding server i. Note that this is a pessimistic estimate as a non–negligible number of these individual combinations are not possible.

3 Critical path

In this section, we present the notion of critical path, which is one of the key ideas allowing to prove and compute the proposed bound.

3.1 Synchronization constraints

The notion of critical path relies on the synchronization constraints of the queueing network, also called evolution equation, or state recursion [4]. This property can be found in [12] for fork–join queueing networks with blocking (with blocking–before–service policy). We present it in the next lemma, in a different form.

To state the synchronization constraints, we first need some notations. As already mentioned, each server is given an index $i = 1, 2, \ldots, m$, with m being the number of servers. The set of servers which directly precede a server i is denoted $E(i)$ and the set of servers which directly follow a server i is denoted $F(i)$. The size of the buffer between servers i and j is denoted $b(i,j)$. Each job is given an index $k = 1, 2, \ldots$, where the first job to leave the system is given index 1. Finally, the moments at which job $w_{i,k}$ starts and ends in the run r are denoted $t^r_{start}(w_{i,k})$ and $t^r_{end}(w_{i,k})$. The time the sample run starts is fixed to zero.

The next lemma gives the synchronization constraints for fork–join queueing networks with blocking, the idea being that a server i can start a job k if three conditions are satisfied. First, each previous server should have finished job k. Second, the previous job on server i, job $k-1$, has to be finished. Third, there should be some room left for this job $k-1$ in each following buffer. In other words, the previous job $k-1$ should not be blocked in server i. Moreover, once all these conditions are satisfied, there is no reason to wait and, so, $w_{i,k}$ starts exactly when the last condition becomes satisfied.

Lemma 1. (Synchronization constraints) *Given a fork–join queueing network, the moment a job $w_{i,k}$ starts, in a sample run r, is given by the following equation:*

$$t^r_{start}(w_{i,k}) = \max\Big[\ \max_{j\in E(i)}\big[\,t^r_{end}(w_{j,k})\,\big], \tag{1}$$

$$t^r_{end}(w_{i,k-1}), \tag{2}$$

$$\max_{j\in F(i)}\big[\,t^r_{start}(w_{j,k-b(i,j)-1})\,\big]\ \ \Big]. \tag{3}$$

Moreover, the starting time of a job is always equal to the ending time of another job, as we have:

$$t^r_{start}(w_{j,k-b(i,j)-1}) = \max\Big[\ t^r_{end}(w_{j,k-b(i,j)-2}), \tag{4}$$

$$\max_{h\in F(j)}\big[\,t^r_{end}(w_{h,k-b(i,j)-b(j,h)-3})\,\big], \tag{5}$$

$$\ldots\ \Big]. \tag{6}$$

Proof. Each term corresponds to one of the conditions mentioned previously, and to one state (starved, working, or blocked) preceding $w_{i,k}$ on server i.

(1) Job $w_{i,k}$ cannot start before each preceding server $j \in E(i)$ passed unit k to server i. If this term corresponds to the maximum, i.e. $t_{start}(w_{i,k}) = \max_{j\in E(i)}[\,t_{end}(w_{j,k})\,]$, it means server i was starved before beginning to work on unit k.

(2) The server i cannot begin to work on unit k before it finished working on the previous unit $k-1$. If $t_{start}(w_{i,k}) = t_{end}(w_{i,k-1})$, it means server i was working just before starting job $w_{i,k}$.

(3) In fact, before starting $w_{i,k}$, server i should not only finish to work on job $w_{i,k-1}$ but should also get rid of unit $k-1$. Therefore, there should be some room left in each following buffer. This is not the case (and server i gets blocked) if units $k-2$ to $k-b(i,j)-1$ are waiting in the queue preceding any server $j \in F(i)$. Indeed, in this case, the buffer is full. So, $w_{i,k}$ can only start when job $k-b(i,j)-1$ began on server j. In this case, i.e. if $t_{start}(w_{i,k})$ equals term (3), it means server i was blocked before beginning to work on unit k.

Once all the above conditions are satisfied, there is no reason to wait. Moreover, equation (1-3) gathers all conditions as every possible state preceding $w_{i,k}$ is considered (starved, working, or blocked).

The second part of the lemma can be directly deduced from equation (1-3). Job $k-b(i,j)-1$ can only start if the previous job $k-b(i,j)-2$ finished (leading to term (4)), and if there is room left in the next buffers for server j to get rid of $k-b(i,j)-2$ (5). The last condition means that $k-b(i,j)-b(j,h)-2$ should have started on server $h \in F(j)$, which means $k-b(i,j)-b(j,h)-3$ should have ended. Similar conditions can then be found for the following servers. □

These synchronization constraints are the basis which allows to define, build and compute the critical path.

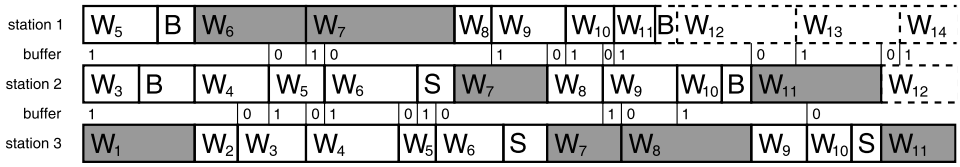

Figure 4: Gantt chart of a sample run, for a three server tandem queue, with buffer sizes $b(1,2) = b(2,3) = 1$. The critical path is given in gray. The time goes from left to right. The state of a server is represented either by a letter (B for blocked, S for starved) or by the job currently served. The state of a queue is represented by the number of units waiting inside.

3.2 Definition and construction

The critical path of the sample run[1] r, denoted $cp(r)$, is defined as the sequence of jobs that covers the run r without gap and without overlap. By definition, the length of a run r (in other words the time to serve the corresponding number of units) can thus be written as a sum of job lengths:

$$l(r) = \sum_{w_{i,k} \in cp(r)} l^r(w_{i,k}). \tag{7}$$

The notion of critical path is illustrated in Figure 4. It can be seen that the critical path allows to cover the sample run by a sequence of jobs.

The critical path can be built quite easily. Starting with the last job that leaves the network (at the end of run r), we look which job end, in this precise run, has triggered its start. This new job will be part of the critical path and, from it, the next job can be found in the same way. Repeating this process, we can proceed backwards in time until the start of the run. As every job start is triggered by the end of another job, every run r has at least one critical path.

Moreover, note that the course of the critical path among the servers can be related to the server state before each job of the path. The predecessor of a job in the critical path can be deduced from the state of the same server just before this job. If the server is previously working, the predecessor is obviously a job on the same server (and this corresponds to the term (2) in Lemma 1). If the server is previously starved, the predecessor is a job on the previous server for which the current server was waiting (term (1), see for

[1]From this point on, we will assume the sample runs to be finite.

example $w_{3,11}$ to $w_{2,11}$ in Figure 4). If the server is previously blocked, the predecessor is a job on the later server which was blocking the current server (terms (4-6), see for example $w_{1,6}$ to $w_{3,1}$ in Figure 4). This characteristic is the key idea which allows us to compute the critical path, as explained in the next section.

3.3 Computation

In this section, we sketch how the critical path can be computed. By computation of the critical path, we mean the computation of the steady–state probabilities $p_\alpha^{cp}(i,s)$ that, at a given discrete time, if the system is in state s, the critical path "lies" on server i . For example, in Figure 3, $p_\alpha^{cp}(1,\mathrm{B12}) = 0$ and $p_\alpha^{cp}(2,\mathrm{B12}) = 1$, as the critical path cannot lie on a blocked server. Note that we use the subscript α because the critical path is, and can only be, computed in the discretized time.

The question is thus how the probabilities $p_\alpha^{cp}(i,s)$ can be computed. They can be computed thanks to the fact that the predecessor of a job in the critical path can be deduced from the state of the same server just before this job (see previous subsection). Let us consider a transition from a system state s to another system state s' and analyze the behavior of the critical path while this transition is encountered. We can infer on which server the critical path will lie in the system state s if we know where *cp* lies in s' and if we know the individual servers states constituting s. The critical path will lie in the (working) server i of s in three cases. First, going backward in time, *cp* will stay on server i in state s if it was already on i in state s' ($w_{3,8}$ to $w_{3,7}$ in Figure 4). Second, *cp* will jump, going backward in time, to a preceding server i in s if it was on server i_f in s', with $i \in E(i_f)$, and if the server was starved by server i previously, i.e. $s_{i_f} = S_i$ ($w_{3,11}$ to $w_{2,11}$ in Figure 4). Third, *cp* will jump, going backward in time, to a following server i in s if it was on server i_e in s', with $i \in F(i_e)$, and if the server was blocked by server i previously, i.e. $s_{i_e} = B_i$ ($w_{2,11}$ to $w_{3,8}$ in Figure 4). To get the probability $p_\alpha^{cp}(i,s)$ that, at a given time, the critical path lies on server i if the system is in state s, we thus simply have to consider these three cases for each possible transition from state s to one of its successors and weight each transition by its probability. We finally get the following

equation, $\forall\, i, s$:

$$
\begin{aligned}
p_\alpha^{cp}(i,s) = \mathbb{1}_{\{s_i \neq S,B\}} \sum_{s' \in \text{ Suc}(s)} \; & p[s \to s'] \Big(p_\alpha^{cp}(i,s') \\
& + \mathbb{1}_{\{s_{i_f} = S_i, i \in E(i_f)\}}\, p_\alpha^{cp}(i_f, s') \\
& + \mathbb{1}_{\{s_{i_e} = B_i, i \in F(i_e)\}}\, p_\alpha^{cp}(i_e, s') \\
& + \mathbb{1}_{\{s_{i_e} = B_i, s_{i_d} = B_{i_e}, i \in F(i_e), i_e \in F(i_d)\}} p_\alpha^{cp}(i_d, s') \\
& + \dots \Big),
\end{aligned}
$$

where $\mathbb{1}_{\{condition\}}$ is the indicator function, i.e. it equals one if the *condition* is satisfied and zero otherwise, $p[s \to s']$ is the transition probability between the system states s and s', and $\text{Suc}(s)$ is the set of successors of s, i.e. $s' \in \text{Suc}(s)$ if $p[s \to s'] > 0$.

Thus, to compute the probabilities $p_\alpha^{cp}(i,s)$, a linear system of equations has to be solved. The number of unknows of this system equals the number of servers m (index i) times the number of system states (index s). In first approximation, its size is thus proportional to $m \cdot \prod_{i=1}^{m}(a_i+2) \cdot \prod_{i=2}^{m}(b_i+1)$ (see Section 2.2). This system of equations has to be solved to compute the bound proposed in Section 4. The complexity is thus the main limitation of the proposed bounding methodology. It can be seen from the formula that the complexity increases when the number of discrete values a increases, in other words when the PMF discretization refines. There is thus a clear trade–off between complexity and accuracy, which can be directly controlled by the parameter a. The size of the Markov chain also quickly increases with the complexity of the system configuration (number of servers and buffer sizes).

3.4 Properties

In this section, we give two properties of the critical path which will be directly used in order to prove the bound in the next section. The first property relates the sequence of jobs defining the critical path in the original (resp. discretized) realisation to the same sequence of jobs in the discretized (resp. original) realisation. The second property relates the number of jobs in the critical path to the number of units served.

We begin with the first property. Let us consider the critical path of a sample run r. It is defined as the sequence of jobs, $cp(r) = \{w_{i,k}\}$, that covers r without gap and without overlap. We would like to know the behavior of this sequence of jobs $\{w_{i,k}\}$ in another run. The equation of Lemma 1 is valid for any run: a job $w_{i,k}$ cannot be started before all the jobs on the

right hand side are finished. This is just a static structural property of the system, independent of the run. Consequently, as two consecutive jobs in the sequence $cp(r) = \{w_{i,k}\}$ satisfy this structural property, in another run, the same sequence of jobs will not show any overlap either. The absence of overlap is independent of the run considered. However, which precise job end will trigger the start of job $w_{i,k}$, i.e. which term of equation (1-3) will be satisfied at equality, depends on the service times and thus on the particular run we consider. In another run, gaps could thus appear between the jobs of the sequence $\{w_{i,k}\}$. In conclusion, while it forms the critical path in the run r, the sequence of jobs $\{w_{i,k}\}$ will just be a non–overlapping path, maybe with gaps, in another run. Obviously, in a given run, the sum of the lengths of the jobs composing a non–overlapping path is always shorter than the length of the critical path, as the first may include gaps.

The second property relates the cardinality $|cp(r_\alpha)|$ of the critical path of the discretized run r_α, i.e. the number of jobs in it, to the number n^r of units served by a server during run r_α. We denote $p_{1,\alpha}$ the steady–state probability that a given server is, at a given time, in first stage of service, in the discretized time. Similarly, we define $p_{1,\alpha}^{cp}$ as the steady–state probability that the server where the critical path lies at a given time is in first stage of service, in the discretized time.

Lemma 2. *The ratio between the number of jobs in the critical path and the number of units served can be computed as follows:*

$$\lim_{n^r \to \infty} \frac{|cp(r_\alpha)|}{n^r} = \frac{p_{1,\alpha}^{cp}}{p_{1,\alpha}}, \qquad \forall\, i.$$

Proof. As, in discrete time, every job passes through its first stage during one time step τ, and as the mean time to serve one job equals, by definition, the cycle time c_α, we have $p_{1,\alpha} = \tau / c_\alpha$. Similarly, as every job of $cp(r_\alpha)$ passes through its first stage during one time step, and if $l_\alpha(w_{cp})$ denotes the average length of a job in the critical path in discrete time, we have $p_{1,\alpha}^{cp} = \tau / l_\alpha(w_{cp})$. Consequently, we get:

$$\frac{p_{1,\alpha}^{cp}}{p_{1,\alpha}} = \frac{c_\alpha}{l_\alpha(w_{cp})} = \lim_{n^r \to \infty} \frac{l(r_\alpha)/n^r}{l(r_\alpha)/|cp(r_\alpha)|} = \lim_{n^r \to \infty} \frac{|cp(r_\alpha)|}{n^r}.$$

□

4 Tight lower bound

At this stage, we are able to show the main result of the paper: a computable and tight lower bound on the cycle time of general fork–join queueing networks. The core idea comes from the first property given in the previous subsection. The sequence of jobs which constitutes the critical path in

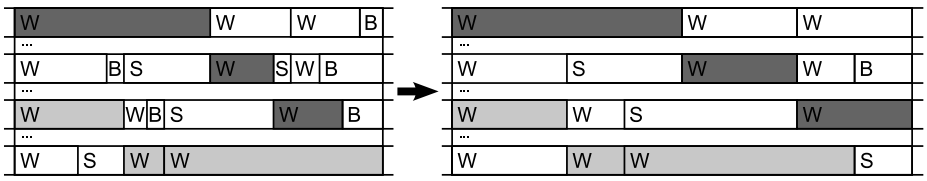

Figure 5: Critical paths and corresponding non–overlapping paths, in an original run (left) and in the discretized run (right, with $\alpha = 0$), for a tandem queue made of four servers.

the discretized time form a non–overlapping path in the original time, thus shorter than the critical path in the original time, and thus shorter than the length of the original sample run. The critical path in the discretized time thus leads to a lower bound in the original time, and, importantly, it can be computed (see Section 3.3).

Before rigorously proving this result, let us illustrate it in Figure 5. The left–hand side shows a run r in the original time and the right–hand side depicts the corresponding discretized run r_α. On the left hand–side, the critical path is colored in light gray. Its length gives the real time to serve three units. On the right–hand side, the critical path is colored in dark gray. The corresponding non–overlapping path is given in dark gray on the left–hand side. It can be seen that it offers a lower bound on the real time to produce three units. In other words, in the original time (left), the dark sequence of jobs is smaller than the light critical path. Moreover, it is reasonable to think that the critical path in the discretized time leads to a non–overlapping path (dark sequence) which is close to the critical path in the original time (light sequence), and thus to a tight lower bound.

In the next proposition, we show that the non–overlapping path in the original time can be computed, and thus, that a good lower bound can be computed. The proof relies on two ideas. First, the probability for a job of the dark sequence to be on a given server and to have its length in a given interval can be deduced from the discrete critical path computation. Second, the expected length of such a job (knowing its discretized length) is independent of the fact that it belongs to the dark sequence and can thus be computed.

Proposition 3. *The proposed modelling method, and the critical path computation, allows to compute the following lower bound on the cycle time c of*

a fork–join queueing network:

$$c \geq \frac{p^{cp}_{1,\alpha}}{p_{1,\alpha}} \sum_{i=1}^{m} \sum_{j=1}^{a} P[l_\alpha(w_{i,k}) = j\tau \,|\, w_{i,k} \in cp(r_\alpha)] \cdot E[l(w_{i,k}) \,|\, l_\alpha(w_{i,k}) = j\tau]. \tag{8}$$

Proof. Let us suppose that we choose a given $w_{i,k}$ in the discrete time (right–hand side of Figure 5) and that its length equals $j\tau$. The only thing we know about the original length of $w_{i,k}$ (left–hand side of Figure 5) is that it lies in the interval for which the probability mass is aggregated on $j\tau$, i.e. between $(j-1)\tau + \alpha$ (0 if $j = 1$) and $j\tau + \alpha$. If $w_{i,k}$ is chosen in the discretized time, independently of its length in the original time, we have no clue about the position of the original length in the interval. The original length (known to be in a given interval $[(j-1)\tau + \alpha, j\tau + \alpha]$) of a job which is chosen because it belongs to the critical path in the discretized time is thus independent of the fact that it belongs to this critical path $cp(r_\alpha)$. We have:

$$E[l(w_{i,k}) \,|\, l_\alpha(w_{i,k}) = j\tau \;\&\; w_{i,k} \in cp(r_\alpha)] = E[l(w_{i,k}) \,|\, l_\alpha(w_{i,k}) = j\tau].$$

Moreover using Lemma 2, the right–hand side term of (8) can be rewritten as follows:

$$\begin{aligned} \lim_{n^r \to \infty} \frac{|cp(r_\alpha)|}{n^r} \sum_{i=1}^{m} \sum_{j=1}^{a} & P[l_\alpha(w_{i,k}) = j\tau \mid\; w_{i,k} \in cp(r_\alpha)] \\ & \cdot E[l(w_{i,k}) \,|\, l_\alpha(w_{i,k}) = j\tau \,\&\, w_{i,k} \in cp(r_\alpha)] \\ & = \lim_{n^r \to \infty} \frac{|cp(r_\alpha)|}{n^r} E[l(w_{i,k}) \,|\, w_{i,k} \in cp(r_\alpha)]. \end{aligned}$$

The expression $|cp(r_\alpha)| \cdot E[l(w_{i,k}) \,|\, w_{i,k} \in cp(r_\alpha)]$ gives the length, in the original time, of the sequence of jobs making the critical path in the discrete time, i.e. the dark sequence on the left-hand side of Figure 5. As it is only a non–overlapping path in the original time, it is shorter than the critical path (in light gray), and thus than the global real time to serve n^r units. Formally, this can be written as follows, and it completes the proof.

$$\lim_{n^r \to \infty} \frac{|cp(r_\alpha)|}{n^r} E[l(w_{i,k}) \,|\, w_{i,k} \in cp(r_\alpha)] \leq \lim_{n^r \to \infty} \frac{|cp(r)|}{n^r} E[l(w_{i,k}) \,|\, w_{i,k} \in cp(r)] = c.$$

□

It is essential to see that this lower bound is computable. Each term can be computed. The two terms implying the critical path $cp(r_\alpha)$ ($p^{cp}_{1,\alpha}$ and $P[l_\alpha(w_{i,k}) = j\tau \,|\, w_{i,k} \in cp(r_{P,\alpha})]$) are easily inferred from the critical path computation, i.e. from the probabilities $p^{cp}_\alpha(i,s)$ (see Section 3.3). We remind that the critical path computation suffers from a high complexity. The probability $p_{1,\alpha}$ is deduced from the steady–state probabilities of the states of the Markov chain (see Section 2.2). Finally, $E[l(w_{i,k}) \,|\, l_\alpha(w_{i,k}) = j\tau]$ can be computed from the original service time distributions, which are known.

It can be argued that the proposed lower bound is tight because the critical path does not differ much from the discretized to the original time.

Moreover, when the number of discretization steps a increases, the length of a discretized job becomes closer to the length of the original job. Consequently, the lower bound becomes tighter when the PMF discretization is refined. At the limit, when a goes to infinity, the discretized time tends to be equivalent to the original time, and the bound thus converges to the exact cycle time. Finally, note that the proposed bound relies on both points presented previously in the paper. First, the critical path, and its computation, is obviously essential here. Second, the lower bound also relies on the idea of probability masses fitting as it uses the fact that the probability mass in an interval is aggregated in one value $j\tau$.

5 Computational experiments

In order to assess the tightness of the proposed bound, and to study its behavior, we compared it to simulations results. Various networks configurations have been tested: tandem, fork and join, with two, three, or four servers (various possibilities in this case). The global storage space of a network goes from zero to four and is supposed to be balanced among the buffers. Each configuration has been tested with various service time distributions, arbitrary chosen among the ten following distributions: uniform(0,1), beta(1.3,1), beta(2,2), beta(4,4), beta(5.5,6), beta(8,8), beta(10,9), triangular(0,1,0.5), triangular(0.2,1,0.3) and triangular(0.1,0.9,0.6). These distributions have various expectations and various coefficients of variation. In total, 700 FJQN have been analyzed. Moreover, the parameters of the modelling method have also been varied aiming to understand their respective influence: the number a of discretization steps (4, 6 and 8 steps) and the shift parameter α ($\alpha/\tau = 0, 0.25, 0.5, 0.75, 1$). In total, we made 10500 experiments. Tables 1 to 3 illustrate the levels of accuracy reached by the lower bound on the cycle time. They give the average relative error, in percent, between the bound and the result of the simulation (i.e. $(sim-bound)/sim$).

Table 1 gives the average relative errors obtained for all the network configurations (tandem, fork, join, with various numbers of servers and various storage spaces). First of all, it shows the tightness of the bound. On average, the level of accuracy reached is 0.4% with eight discretization steps, 0.7% with six and 1.4% with only four steps. We remind that this accuracy, as well as the bound caracteristic, is valid for general distributions, i.e. for the global modelling process, including the building of tractable distributions. In Table 1, the influence of parameters a and α is also illustrated. It can be seen that the tightness of the bound increases significantly when the

α/τ	0	0.25	0.5	0.75	1
$a = 4$	1.74%	1.53%	1.41%	**1.37%**	1.41%
$a = 6$	0.83%	0.74%	0.68%	**0.64%**	0.64%
$a = 8$	0.50%	0.45%	0.42%	**0.39%**	0.39%

Table 1: Average bound tightness reached on all FJQN networks configurations. The shift parameter α and the number a of discretization steps vary.

discretization is refined. This is not surprising as the approximation of the distributions improves, so does the approximation of the critical path. The shift parameter α has less impact. In the next tables, we choose to focus on the case where the mass is grouped in the middle of the interval ($\alpha/\tau = 0.5$), which is the most natural. Other parameters α lead to the same behavior.

Table 2 aims to illustrate the influence of the configuration of the network on the tightness of the bound. It can be seen that the accuracy reached for tandem, fork or join networks is very similar. The number of servers composing the network has a significant influence. Nevertheless, the bound stays tight for larger networks. Table 2 also gives the size of the system of equations to be solved (note that the matrix of the system is sparse and structured). It can be seen that it quickly increases with the number a of discretization steps and with the size of the system. In terms of computa-

	Tandem			Fork		Join	
m	2	3	4	3	4	3	4
$a = 4$	0.83%	1.30%	1.64%	1.27%	1.55%	1.30%	1.62%
	84	*558*	*2424*	*618*	*2656*	*618*	*2904*
$a = 6$	0.38%	0.61%	0.80%	0.61%	0.75%	0.64%	0.79%
	176	*1530*	*7852*	*1728*	*8884*	*1728*	*9612*
$a = 8$	0.21%	0.36%	0.51%	0.36%	0.48%	0.38%	0.51%
	300	*3186*	*18316*	*3630*	*21276*	*3630*	*22572*

Table 2: Average bound tightness reached for various FJQN networks configurations, with various numbers of servers m. The number a of discretization steps varies, and $\alpha/\tau = 0.5$. The size of the linear system of equations to be solved is given in italic (for $\sum b(i,j) = 2$).

$\sum b(i,j)$	0	1	2	3	4
$a = 4$	1.98%	1.78%	1.38%	1.03%	0.87%
	120	*246*	*558*	*870*	*1374*
$a = 6$	0.98%	0.87%	0.66%	0.49%	0.40%
	258	*594*	*1530*	*2466*	*4071*
$a = 8$	0.58%	0.51%	0.39%	0.29%	0.17%
	444	*1122*	*3186*	*5250*	*8940*

Table 3: Average bound tightness reached for various storage spaces. The number a of discretization steps varies, and $\alpha/\tau = 0.5$. The size of the linear system of equations to be solved is given in italic (for a three station line).

tional time[2], the method (construction and resolution) takes 0.2, 1.7 and 40 seconds to compute the bound, for tandem queues with 2, 3 and 4 servers, respectively, and with $a = 8$ and $\sum b(i,j) = 2$.

Table 3 shows that the bound tends to tighten when the storage space of the network is increased. In summary, concerning the configuration of the network, while the type of configuration does not seem to matter, the tightness of the bound seems to deteriorate when the number of servers increases, and it seems to improve when the storage space increases.

Ideally, these results should be compared to those of concurrent methods. However, as explained in the introduction, few methods exist. The contribution of Baccelli [4] is mainly theoretical. To the best of our knowledge, the accuracy of the results have not been investigated. In previous work, we proposed another bounding methodology, based on the result of the modelling methodology, without taking advantage of the critical path computation (see Section 2.1 or [20] for more details). The tightness of the bounds is directly related to the step size τ, and is clearly not as good. For example, for three station tandem queues, with $a = 8$ and $\alpha/\tau = 0.5$, the average relative error equals 9.9%. This can be compared to the 0.36% tightness reached with the present bound (see Table 2).

[2]Using the Gaussian elimination implemented in MATLAB® on a 2.16 GHz usual PC, 2 GB RAM.

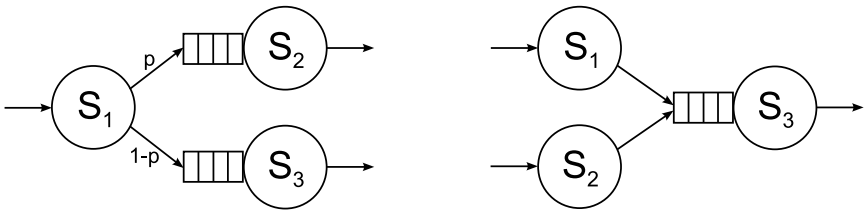

Figure 6: Examples of three station split (left) and merge (right) networks.

6 Extensions

6.1 Split and merge configurations

In this paper, we proposed a new bounding methodology, which allows to compute a tight lower bound on the cycle time. For the sake of readability, we described it for fork–join queueing networks, for which the description is the simplest. However, the methodology could be applied or extended to other queueing networks.

In this subsection, we are interested in split and merge queueing networks with blocking. In such networks, the nodes are linked arbitrarily without forming loops, i.e. the servers can have several input or output servers. Unlike in FJQN, one buffer is associated to each server, and a server exclusively takes items from this buffer. In split and merge networks, items are not assembled or disassembled, one item entering the system leads to exactly one unit leaving the system. In a split configuration, the item is routed to one of the following servers, with some routing probabilities (see Figure 6). In a merge configuration, the merge server takes the items from the buffer, which gathers the items from every preceding server (see Figure 6). Like in the rest of the paper, we suppose general finite service time distributions, saturation of the network, and "blocking after service" policy.

The bounding methodology can be straightforwardly extended to split configurations. The critical path in the discretized run leads to a non–overlapping path in the original time and thus to a lower bound. In order to assess the tightness of this bound, we compute it for 220 split configurations, similarly to the experiments reported in Section 5. In order to keep the network balanced, some distributions, for split servers notably, were shrunk to a domain which is half of the previous one ([0 0.5] instead of [0 1]). The results are summarized in Table 4. They show the tightness of the bound for split networks. The average accuracy reached are similar but slightly less

m	3	4
$a = 4$	1.81%	2.32%
	501	*2324*
$a = 6$	0.70%	0.98%
	1434	*8328*
$a = 8$	0.38%	0.57%
	3045	*21024*

Table 4: Average bound tightness reached for the split configurations, with various numbers of servers m. The number a of discretization steps varies, and $\alpha/\tau = 0.5$. The size of the linear system of equations to be solved is given in italic (with $\sum b(i,j) = 2$).

good than the one reached on previously studied networks (see Table 2). This can be explained by the fact that some distributions are shrunk and thus less finely discretized (these distributions are discretized to 3 values when $a = 6$ for example).

Concerning merge configurations, the extension of the bounding methodology turns out to be trickier, if not infeasible. Indeed, with merge servers, a key property of the critical path does not hold anymore: the sequence of jobs which makes the critical path in the discrete run is not necessarily non–overlapping in the original run. If a merge server is starved, the preceding server from which the unstarving job comes is the first to finish[3]. It thus depends on the considered run: the unstarving server could be different from the discrete to the original run. In such case, as illustrated in Figure 7, an overlap can appear in the sequence of job, in the original run. The bounding methodology can thus not be directly extended to merge configurations. Moreover, it seems to be difficult to bound the overlap lengths or to find another (tight) non–overlapping path.

6.2 Approximations

In the core of the paper, we focus on showing a bound on the productivity, which is the only exact information reachable with general service time distributions. However, the performances can also be evaluated from the modelling method described in Section 2. A straightforward approximation of the cycle time is given by the cycle time of the system in the discretized

[3]The structural constraints (see Lemma 1) are changed. The first term (1) is replaced by $\min_{j \in E(i)}[\, t^r_{end}(w_{j,k'})\,]$.

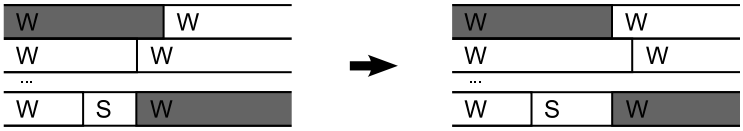

Figure 7: Example, illustrated on a Gantt chart, where the sequence of job defining the critical path, colored in dark gray, in the discretized run (right) leads to an overlap in the original run (left).

time (with $\alpha/\tau = 0.5$), which can be computed from the Markov chain. Of course, the bounding property is lost in these estimation. In this section, we briefly illustrate this other feature of the proposed modelling method. The accuracy of the cycle time approximation is studied on various network configurations and compared to simulation results.

A total of 700 fork–join queueing networks were tested. The configurations, the service time distributions and the storage spaces are the same as for the bound tests (see Section 5). Table 5 shows the good accuracy of the approximation. On average, the level of accuracy reached is approximately 0.13% with eight discretization steps, 0.22% with six and 0.5% with only four steps. Moreover, the accuracy of the approximation turns out to be remarkably stable with the configuration and the number of servers. Table 6 shows that the accuracy is also stable in the storage space, even if it seems to be slightly deteriorating. These tables can be compared to Tables 2 and 3

	Tandem			Fork		Join	
m	2	3	4	3	4	3	4
$a = 4$	0.54%	0.50%	0.43%	0.48%	0.46%	0.44%	0.49%
	42	*186*	*606*	*206*	*664*	*206*	*726*
$a = 6$	0.24%	0.22%	0.19%	0.21%	0.21%	0.20%	0.22%
	88	*510*	*1963*	*576*	*2221*	*576*	*2403*
$a = 8$	0.13%	0.13%	0.12%	0.12%	0.13%	0.12%	0.13%
	150	*1062*	*4579*	*1210*	*5319*	*1210*	*5643*

Table 5: Average accuracy of the approximation for various fork–join configurations, with various numbers of servers m. The number a of discretization steps varies, and $\alpha/\tau = 0.5$. The size of the linear system of equations to be solved is given in italic (for $\sum b(i,j) = 2$).

$\sum b(i,j)$	0	1	2	3	4
$a = 4$	0.45%	0.41%	0.46%	0.51%	0.55%
	40	*83*	*186*	*290*	*458*
$a = 6$	0.21%	0.19%	0.21%	0.23%	0.23%
	86	*198*	*510*	*822*	*1357*
$a = 8$	0.12%	0.12%	0.12%	0.13%	0.15%
	148	*374*	*1062*	*1750*	*2980*

Table 6: Average accuracy of the approximation on FJQN, for various storage spaces. The number a of discretization steps varies, and $\alpha/\tau = 0.5$. The size of the linear system of equations to be solved is given in italic (for a three station line).

giving the tightness of the bound. It reveals that the approximation is more accurate, which is not surprising as the bounding property is lost. However, this does not seem to be true when the storage space increases. When the buffer sizes increase, the bound seems to become as accurate as the approximation (see Tables 3 and 6). From these tables, it can also be observed that the complexity of the computation of the bound is higher than the one of the approximation.

Finally, the cycle time approximation has been tested on 220 split networks and 220 merge networks (with the service time distributions previously mentioned). Table 7 reveals that the accuracy of the approximation

	Split		Merge	
m	3	4	3	4
$a = 4$	1.69%	1.21%	2.94%	2.48%
	167	*581*	*306*	*726*
$a = 6$	0.59%	0.46%	0.95%	0.92%
	478	*2082*	*576*	*2403*
$a = 8$	0.30%	0.22%	0.43%	0.44%
	1015	*5256*	*1210*	*5643*

Table 7: Average accuracy of the approximation for various split–and–merge configurations, with various numbers of servers m. The number a of discretization steps varies, and $\alpha/\tau = 0.5$. The size of the linear system of equations to be solved is given in italic (for $\sum b(i,j) = 2$).

is good, even if it is not as good as for FJQN. From these results, it can be said that the modelling method can be satisfyingly used for split–and–merge queueing networks too.

7 Conclusion

In this paper, we proposed a computable, tight and general lower bound on the cycle time of queueing networks with blocking and with general distributions. The methodology is presented for fork–join queueing networks and extended to split configurations. As the service time distributions are general, an exact analysis is impossible. A bound is thus the only exact information available.

In a few words, the bounding methodology works as follows. The distributions are discretized by probability masses fitting, i.e. the probability masses on regular intervals are aggregated on a particular value of the corresponding interval. With the discretized distributions, the critical path, a sequence of jobs which covers the run, can be computed. In the original run, this sequence of jobs is non–overlapping, shorter than the run, and thus leads to a lower bound on the cycle time. Moreover, it can be supposed that this sequence is close to the critical path in the original time, and that this becomes more and more true when the discretization is refined, i.e. the bound becomes tighter and tighter.

In order to assess the tightness of the bound, we compared the results of the method to simulation results. It showed the good accuracy of the methodology : the average relative error equals 1.4%, 0.7% and 0.4% with 4, 6 and 8 discretization steps, for fork–join queueing networks. Note that there is a trade–off between the accuracy and the complexity, which is the main weakness of the approach. We also argued that the methodology can be straightforwardly extended to split configurations and it leads to similar tightness. Finally, we showed that the approximation of the cycle time, given by the cycle time of the queueing network (fork–join or split–and–merge) with the discretized distributions, is accurate (0.5%, 0.2% and 0.13% with 4, 6 and 8 discretization steps, for FJQN).

References

[1] T. Altiok. *Performance Analysis of Manufacturing Systems*. Springer, New York, 1996.

[2] S. Asmussen, A. Nerman, and M. Olsson. Fitting phase-type distributions via the em algorithm. 23:419–441, 1996.

[3] F. Baccelli and Z. Liu. Comparison properties of stochastic decision free petri nets. *IEEE Trans. On Automatic Control*, 37:1905–1920, 1992.

[4] F. Baccelli and A.M. Makowski. Queueing models for systems with synchronization constraints. *Proceedings of the IEEE*, 77:138–161, 1989.

[5] S. Balsamo, V. de Nitto Personé, and R. Onvural. *Analysis of Queueing Networks with Blocking*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.

[6] A. Bobbio, A. Horváth, M. Scarpa, and M. Telek. Acyclic discrete phase type distributions : Properties and a parameter estimation algorithm. *Performance Evaluation*, 54:1–32, 2003.

[7] A. Bobbio, A. Horváth, and M. Telek. Matching three moments with minimal acyclic phase type distributions. 21:303–326, 2005.

[8] J.A. Buzacott and J.G. Shanthikumar. *Stochastic Models of Manufacturing Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[9] P.-J. Courtois and P. Semal. Computable bounds for conditional steady-state probabilities in large markov chains and queueing models. *IEEE Journal on Selected Areas in Communications*, 4(6):926–937, 1986.

[10] Y. Dallery and Y. Frein. On decomposition methods for tandem queueing networks with blocking. *Operations Research*, 41(2):386–399, 1993.

[11] Y. Dallery and S.B. Gershwin. Manufacturing flow line systems: a review of models and analytical results. *Queueing Systems*, 12:3–94, 1992.

[12] Y. Dallery, Z. Liu, and D. Towsley. Equivalence, reversibility, symmetry and concavity properties in fork/join queueing networks with blocking. *Journal of the Association for Computing Machinery*, 41:903–942, 1994.

[13] D.L. Eager and K.C. Sevcik. Bound hierarchies for multiple-class queuing networks. *Journal of the Association for Computing Machinery*, 33(1):179–206, 1986.

[14] M.A. Johnson and M.R. Taaffe. Matching moments to phase distributions: Mixtures of erlang distributions of common order. 5(4):711–743, 1989.

[15] L. Kerbache and J. MacGregor Smith. The generalized expansion method for open finite queueing networks. *European Journal of Operational Research*, 32:448–461, 1987.

[16] S. Kumar, R. Srikant, and P.R. Kumar. Bounding blocking probabilities and throughput in queueing networks with buffer capacity constraints. *Queueing Systems*, 28:55–77, 1998.

[17] A. Lang and J.L. Arthur. Parameter approximation for phase-type distributions. In S.R. Chakravarthy and A.S. Alfa, editors, *Matrix Analytical Methods in Stochastic Models*, pages 151–206, New York, 1997. Marcel Dekker.

[18] X.-G. Liu and J.A. Buzacott. A decomposition-related throughput property of tandem queueing networks with blocking. *Queueing Systems*, 13.

[19] H.G. Perros. *Queueing Networks with Blocking : Exact and Approximate Solutions*. Oxford University Press, 1994.

[20] J.-S. Tancrez, P. Chevalier, and P. Semal. Probability masses fitting in the analysis of manufacturing flow lines. Core Discussion Paper 2008/28, Université catholique de Louvain, 2008. Submitted to Annals of Operations Research.

[21] J.-S. Tancrez, P. Semal, and P. Chevalier. Histogram based bounds and approximations for production lines. *European Journal of Operational Research*, in press.

[22] P. Tsoucas and J. Walrand. Monotonicity of throughput in non-markovian networks. *Journal of Applied Probability*, 26:134–141, 1989.

[23] J. M. Varah. On fitting exponentials by nonlinear least squares. *SIAM Journal on Scientific and Statistical Computing*, 6(1):30–44, 1985.

[24] W. Whitt. Approximating a point process by a renewal process, i: Two basic methods. *Operations Research*, 30(1):125–147, 1982.

Recent titles

CORE Discussion Papers

2008/25. Ana MAULEON, Vincent VANNETELBOSCH and Cecilia VERGARI. Market integration in network industries.
2008/26. Leonidas C. KOUTSOUGERAS and Nicholas ZIROS. Decentralization of the core through Nash equilibrium.
2008/27. Jean J. GABSZEWICZ, Didier LAUSSEL and Ornella TAROLA. To acquire, or to compete? An entry dilemma.
2008/28. Jean-Sébastien TRANCREZ, Philippe CHEVALIER and Pierre SEMAL. Probability masses fitting in the analysis of manufacturing flow lines.
2008/29. Marie-Louise LEROUX. Endogenous differential mortality, non monitored effort and optimal non linear taxation.
2008/30. Santanu S. DEY and Laurence A. WOLSEY. Two row mixed integer cuts via lifting.
2008/31. Helmuth CREMER, Philippe DE DONDER, Dario MALDONADO and Pierre PESTIEAU. Taxing sin goods and subsidizing health care.
2008/32. Jean J. GABSZEWICZ, Didier LAUSSEL and Nathalie SONNAC. The TV news scheduling game when the newscaster's face matters.
2008/33. Didier LAUSSEL and Joana RESENDE. Does the absence of competition *in* the market foster competition *for* the market? A dynamic approach to aftermarkets.
2008/34. Vincent D. BLONDEL and Yurii NESTEROV. Polynomial-time computation of the joint spectral radius for some sets of nonnegative matrices.
2008/35. David DE LA CROIX and Clara DELAVALLADE. Democracy, rule of law, corruption incentives and growth.
2008/36. Jean J. GABSZEWICZ and Joana RESENDE. Uncertain quality, product variety and price competition. 2008/37. Gregor ZOETTL. On investment decisions in liberalized electricity markets: the impact of price caps at the spot market.
2008/38. Helmuth CREMER, Philippe DE DONDER, Dario MALDONADO and Pierre PESTIEAU. Habit formation and labor supply.
2008/39. Marie-Louise LEROUX and Grégory PONTHIERE. Optimal tax policy and expected longevity: a mean and variance approach.
2008/40. Kristian BEHRENS and Pierre M. PICARD. Transportation, freight rates, and economic geography.
2008/41. Gregor ZOETTL. Investment decisions in liberalized electricity markets: A framework of peak load pricing with strategic firms.
2008/42. Raouf BOUCEKKINE, Rodolphe DESBORDES and Hélène LATZER. How do epidemics induce behavioral changes?
2008/43. David DE LA CROIX and Marie VANDER DONCKT. Would empowering women initiate the demographic transition in least-developed countries?
2008/44. Geoffrey CARUSO, Dominique PEETERS, Jean CAVAILHES and Mark ROUNSEVELL. Space-time patterns of urban sprawl, a 1D cellular automata and microeconomic approach.
2008/45. Taoufik BOUEZMARNI, Jeroen V.K. ROMBOUTS and Abderrahim TAAMOUTI. Asymptotic properties of the Bernstein density copula for dependent data.
2008/46. Joe THARAKAN and Jean-Philippe TROPEANO. On the impact of labor market matching on regional disparities.
2008/47. Shin-Huei WANG and Cheng HSIAO. An easy test for two stationary long processes being uncorrelated via AR approximations.
2008/48. David DE LA CROIX. Adult longevity and economic take-off: from Malthus to Ben-Porath.
2008/49. David DE LA CROIX and Gregory PONTHIERE. On the Golden Rule of capital accumulation under endogenous longevity.
2008/50. Jean J. GABSZEWICZ and Skerdilajda ZANAJ. Successive oligopolies and decreasing returns.
2008/51. Marie-Louise LEROUX, Pierre PESTIEAU and Grégory PONTHIERE. Optimal linear taxation under endogenous longevity.

Recent titles

CORE Discussion Papers - continued

2008/52. Yuri YATSENKO, Raouf BOUCEKKINE and Natali HRITONENKO. Estimating the dynamics of R&D-based growth models.
2008/53. Roland Iwan LUTTENS and Marie-Anne VALFORT. Voting for redistribution under desert-sensitive altruism.
2008/54. Sergei PEKARSKI. Budget deficits and inflation feedback.
2008/55. Raouf BOUCEKKINE, Jacek B. KRAWCZYK and Thomas VALLEE. Towards an understanding of tradeoffs between regional wealth, tightness of a common environmental constraint and the sharing rules.
2008/56. Santanu S. DEY. A note on the split rank of intersection cuts.
2008/57. Yu. NESTEROV. Primal-dual interior-point methods with asymmetric barriers.
2008/58. Marie-Louise LEROUX, Pierre PESTIEAU and Gregory PONTHIERE. Should we subsidize longevity?
2008/59. J. Roderick McCRORIE. The role of Skorokhod space in the development of the econometric analysis of time series.
2008/60. Yu. NESTEROV. Barrier subgradient method.
2008/61. Thierry BRECHET, Johan EYCKMANS, François GERARD, Philippe MARBAIX, Henry TULKENS and Jean-Pascal VAN YPERSELE. The impact of the unilateral EU commitment on the stability of international climate agreements.
2008/62. Giorgia OGGIONI and Yves SMEERS. Average power contracts can mitigate carbon leakage.
2008/63. Jean-Sébastien TANCREZ, Philippe CHEVALIER and Pierre SEMAL. A tight bound on the throughput of queueing networks with blocking.

Books

Y. POCHET and L. WOLSEY (eds.) (2006), *Production planning by mixed integer programming.* New York, Springer-Verlag.
P. PESTIEAU (ed.) (2006), *The welfare state in the European Union: economic and social perspectives.* Oxford, Oxford University Press.
H. TULKENS (ed.) (2006), *Public goods, environmental externalities and fiscal competition.* New York, Springer-Verlag.
V. GINSBURGH and D. THROSBY (eds.) (2006), *Handbook of the economics of art and culture.* Amsterdam, Elsevier.
J. GABSZEWICZ (ed.) (2006), La différenciation des produits. Paris, La découverte.
L. BAUWENS, W. POHLMEIER and D. VEREDAS (eds.) (2008), *High frequency financial econometrics: recent developments*. Heidelberg, Physica-Verlag.
P. VAN HENTENRYCKE and L. WOLSEY (eds.) (2007), *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems*. Berlin, Springer.

CORE Lecture Series

C. GOURIÉROUX and A. MONFORT (1995), Simulation Based Econometric Methods.
A. RUBINSTEIN (1996), Lectures on Modeling Bounded Rationality.
J. RENEGAR (1999), A Mathematical View of Interior-Point Methods in Convex Optimization.
B.D. BERNHEIM and M.D. WHINSTON (1999), Anticompetitive Exclusion and Foreclosure Through Vertical Agreements.
D. BIENSTOCK (2001), Potential function methods for approximately solving linear programming problems: theory and practice.
R. AMIR (2002), Supermodularity and complementarity in economics.
R. WEISMANTEL (2006), Lectures on mixed nonlinear programming.